ARMY RESEARCH LABORATORY

Gerber Format to CNC Punch Translator

Kenneth J. Keyes

ARL-TR-418 August 1994

NOTICES

Disclaimers

The findings in this report are not to be construed as an official Department of the Army position, unless so designated by other authorized documents.

The citation of trade names and names of manufacturers in this report is not to be construed as official Government endorsement or approval of commercial products or services referenced herein.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	August 1994	Technical Report

4. TITLE AND SUBTITLE
GERBER FORMAT TO CNC PUNCH TRANSLATOR

5. FUNDING NUMBERS
C: DAAL01-92-C-0226

6. AUTHOR(S)
Kenneth J. Keyes*

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
US Army Research Laboratory (ARL)
Electronics and Power Sources Directorate (EPSD)
ATTN: AMSRL-EP-I
Fort Monmouth, NJ 07703-5601

8. PERFORMING ORGANIZATION REPORT NUMBER
ARL-TR-418

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES
*Kenneth J. Keyes (Vitronics, Inc., 15 Meridian Rd, Eatontown, NJ 07724) performed this study at the Army Research Laboratory for the project coordinator, Dr. Michael Dukes, of the Electronics and Power Sources Directorate.

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution is unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

Application specific translators provide an efficient means of converting the output of an engineering software package to match the input of another engineering software package; thus, enabling an automated conversion between engineering software packages. In this report, one such application of a translator which provides a vehicle to program a CNC Punch Machine (an automated hole punching machine) from a Gerber Format File (a standard format for PCB artwork photoplotting) is presented. Such a translator, called the Gerber Format to CNC Punch Machine Translator, aids in the design of low temperature co-fired (LTCC) Multi-Chip Modules (MCMs). The Gerber Format to CNC Punch Machine Translator enables the MCM designer to program the CNC Punch Machine to punch via holes, using the same Gerber file which is used to photoplot the MCM signal traces. In doing so, a mechanical engineering package which would normally be used to program the CNC Punch Machine is completely eliminated from the MCM design process.

14. SUBJECT TERMS	15. NUMBER OF PAGES
Computer automated engineering; multichip module; automated micropunch; translator; Gerber format	23
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UL

Table of Contents

1.0 Introduction

The Electronics and Power Sources Directorate (EPSD) of the Army Research Laboratory has in-house capability of prototyping miniaturized systems based on Multi-Chip Modules (MCMs) using a low temperature co-fired ceramic (LTCC) process. A machine used in the process of making LTCC MCMs is the CNC Punch Machine (an automated hole punching machine). The CNC Punch is used to punch holes for the vias (a feed-thru which connects signal traces on different layers) in the MCM. Previously, the CNC Punch Machine had to be programmed manually. The X and Y coordinates for each via hole to be punched had to be entered by hand from the keyboard. This proved to be rather tedious for large MCM designs with several vias and multiple hole sizes.

In this technical report, a vehicle which automates the generation of X and Y coordinates for holes punched by the CNC Punch Machine is presented. Such a vehicle is a translator from Gerber Word Format to CNC Punch Machine File format (developed at the Electronics and Power Sources Directorate of the Army Research Laboratory). The Gerber to CNC Punch Translator reads a gerber file (a standard format for PCB artwork photoplotting) and generates files of X and Y coordinates which are used to program the CNC Punch Machine. The semantics of the Gerber Format and the CNC Punch Machine File format will be discussed. Second, the steps involved in translating a gerber file to a CNC Punch Machine file will be discussed. Finally an example of an MCM design is presented from beginning to end.

2.0 Gerber Format

Gerber Format is a versatile output file used by most Computer-aided Design (CAD) systems which generate photoplots for printed circuit board artwork. The Gerber Format is used for automated drilling, routing, tool selection and insertion machines. This report will only address the area of Gerber Format used by Intergraph (an in-house tool at EPSD for creating MCM artwork) to generate via coordinates on an MCM, mentioning the full capability of Gerber Format only for completeness.

2.1 Preparatory Function Codes

Preparatory function words consist of the address character G followed by two digits. G codes define how an entire block of data is to be processed. Two of the G codes which have significant meaning to Intergraph when creating an MCM are G04 and G54. G04 is used to designate a comment field in the gerber file. G54 is a tool selection code and is used in conjunction with D codes.

2.2 Draft Codes

Draft codes consist of the address character D, followed by two digits to select and control tools. When used with a G54 code, an aperture tool between D10 and D255 is selected. D codes between D01 and D03 are tool control codes for tool up, tool down and flash. When generating via coordinates for an MCM, Intergraph only uses the D03 code. The CNC Punch recognizes the D03 code as a punch hole command.

2.3 Miscellaneous Codes

Miscellaneous codes consist of the address character M, followed by two digits to control system scale factor and perform miscellaneous functions. Typically, only the M02 code is used by Intergraph to signify end of file.

2.4 X-Y Axis Coordinate

X-Y coordinate commands define the point to which the plotting tool must be moved.

2.5 Gerber Format Word Example

Shown below is a gerber file for a design with four holes, spaced 0.30 inch apart forming a square. Following Gerber Format, the gerber file begins on line one with a tool select command, G54, selecting aperture tool D19. Lines two thru five are flash commands at the given X and Y coordinates. It is important to notice that there is no X coordinate on lines three and five. The X coordinate did not change with the new Y coordinate; therefore, the X coordinate was not repeated in the following gerber words on lines three and five. Gerber words are ended, or separated, with an asterisk (*). The M02 command on line six designates end of file.

```
G54D19*
X1000Y1000D03*
Y4000D03*
X4000Y1000D03*
Y4000D03*
M02*
```

3.0 CNC Punch Machine Format

The CNC Punch is an automated machine for punching holes in a piece of material. Up to four different sized punch tools can be loaded at a time, requiring four sets of X and Y coordinates, before having to retool the machine with new sized punches. Hole punching is done in blocks, the punch machine completes punching all of the coordinates of a block before selecting the next punch tool. A block of X and Y coordinates can be either manually entered in at the keyboard or loaded from ASCII files which are grouped by hole size. Based from a programmable origin, the CNC punch machine punches holes at coordinates given in absolute mode and in inches. There are three options for entering a pattern definition: as an X-Y coordinate, as a line, and as an area. This report will only address the X-Y coordinate definition, since this is the mode used at EPSD.

3.1 X-Y Coordinate Pattern

The X-Y coordinate method is the simplest way to program a pattern for the CNC Punch machine. A block of X and Y coordinates is created in the format shown below. The pattern is limited to 3800 holes for each block. Finally, there must be a separate block of X and Y coordinates for each size hole to be punched.

X	Y	
+0.0000	-0.0000	Minimum values in inches
+9.9999	-9.9999	Maximum values in inches

The first character in each column must be either + or - sign.

The second character can be any number between 0 and 9.

The third character is always a decimal point.

The four digits to the right of the decimal point can be any number to 0000 to 9999. The software requires that all four digits be entered, even if they are all zero's.

The "X" coordinate is always followed by a space, then the "Y" coordinate is presented in the same manner as the "X" coordinate.

4.0 Translating Gerber Format to CNC Punch Format

A rule-based language, Prolog, is used to completely write the Gerber to CNC Punch Translator. Based on the first character of each control word in the gerber file, a series of Prolog rules are executed to map the control word into a Prolog intermediate form. Upon completion of reading the gerber file, the intermediate file is then parsed. As such, Prolog mapping rules are constructed to transform the intermediate form into a series of files in CNC Punch format. The files created are files of X and Y coordinates representing blocks of via holes to be punched.

4.1 The Gerber to CNC Punch Translator Program

A runtime entry point is provided. This is done so that a stand-alone program can be created. The definition of a stand-alone program is a Prolog program that can be executed independent of the Prolog Development System. In essence, a run time system is created which can be distributed to a system that does not host the Prolog Development System.

```
runtime_entry(start) :- st.
```

The Gerber to CNC Punch Machine Translator program starts with the rule **st/0**. The user is prompted to enter in the name of the gerber file. Note that single quotations must be put around the entered file name. When Prolog reads an entry from the keyboard, it treats the text input as a Prolog atom. File names with an extension (.ext) can potentially cause problems for Prolog because of the significance of the period. In Prolog the period is used to designate the end of a rule, or series of rules, or the end of input. By placing single quotes around the filename, it will force Prolog to recognize file names, with ".ext", as an atom, thus enabling Prolog to open the gerber file for reading. This section also opens a Prolog intermediate file which aids in the translation of Gerber to CNC Punch format.

```
st :-
        write('[ Enter the gerber file with extension to be translated ]'),
        nl,
        write('[ surrounded by single quotations followed by a "."   ]'),
        nl,
        write('[ example  "file.ext".                                ]'),
        nl,
        read(Filename),
        see(Filename),
        tell(gerberint),
        trans_gerber(0).
```

These following two Prolog rules signal the translator to either continue on or to close the gerber input file and to jump down to the next phase of the translator. If continuing on, the translator reads the gerber file character by character and uses the rule **interput_char/1** to

extract the gerber control words. If continuing to the next phase, the intermediate file is closed for output, the gerber file is closed for reading, and the translator jumps to start2.

```
trans_gerber(-1) :-
        seen,
        told, start2.

trans_gerber(_) :-
        get(X),
        interput_char(X).
```

The leading character of a gerber word is interpreted by **interput_char/1**. If the leading character is an "X", "Y" or "D", "G" or "M" the line is translated to an X-Y coordinate. If the leading character is a "G", then an interpretation of comment or tool selection is executed. All other "G" codes are not recognized by the punch machine and are skipped over, allowing the translation to continue. An "M" code signifies end of file and a "-1" is Prolog's way of knowing "past-end-of-file". When Prolog reads in a character it is represented in decimal ASCII form. The numbers in each rule represent the ASCII code for each leading character of a gerber word, as ordered above.

```
interput_char(88) :-
        write('x('),
        get_values.

interput_char(89) :-
        write('y('),
        get_values.

interput_char(68) :-
        write('d('),
        get_values.

interput_char(71) :-
        write('g('),
        get(X),
        process(X),
        get(X1),
        process(X1),
        get(X2),
        determine_g_code(X2).

interput_char(77) :-
        write('m('),
        get_values, !.

interput_char(-1) :-
        trans_gerber(-1).
```

The Prolog rule **determine_g_code/1** is called by **interput_char/1** to continue extracting the "G" code from the gerber file. Based on the rules below, the current character in the gerber word is determined to be either a "D" or an asterisk (*). If the current character is a "D", "D" code being representative of an aperture tool selection, a comma is written to the intermediate file. The translator then moves on to extract the aperture tool from the gerber file. If the current character is an asterisk, representing end of gerber word, the translator ends mapping the gerber word in the intermediate form and moves on to extract the next gerber word from the gerber file.

```
determine_g_code(68) :-
        write(',),
        get_values, !.

determine_g_code(42) :-
        write(').'),
        nl,
        trans_gerber(0), !.
```

If the "G" code was determined to be a comment (G04), then the characters in the comment field are read until the end of gerber word character (*) is detected. All characters in a comment field are disregarded, for comments are not translated. The following rule enables the translator to continue on when comment fields exist in a gerber file.

```
determine_g_code(_X) :-
        get(X),
        determine_g_code(X).
```

After a determination of the leading character of a new gerber word (e.g., X,Y,D,G,M)is done, the remaining characters of the gerber word are read in and translated to the intermediate file in Prolog format. A Prolog structure is created for each gerber word.

```
get_values :-
        get(X),
        process(X),
        get_values.
```

The following Prolog rules, **process/1**, recognize the ASCII code in decimal for the numbers 0 thru 9 and the letters "X", "Y", "D" and "*". When Prolog performs a **get/1** character, get(X), the character is represented in its ASCII decimal form.

```
process(88) :-
        write('x(').

process(89) :-
```

```
        write(',').

process(45) :-
        write('-').

process(48) :-
        write('0').

process(49) :-
        write('1').

process(50) :-
        write('2').

process(51) :-
        write('3').

process(52) :-
        write('4').

process(53) :-
        write('5').

process(54) :-
        write('6').

process(55) :-
        write('7').

process(56) :-
        write('8').

process(57) :-
        write('9').
```

ASCII code in decimal for a "D".

```
process(68) :-
        write(',').
```

ASCII Code in decimal for "*".

```
process(42) :-
        write(').'),
        nl,
        trans_gerber(0).
```

Phase two of the translator begins with **start2/0**. The intermediate file, gerberint, is read to convert the X-Y coordinates to a format that the punch machine can recognize. In a gerber file, if the X-coordinate or the Y-coordinate does not change in the next gerber word, then that coordinate is not repeated in the gerber file. Therefore, this translator must keep track of the previous X-Y coordinate so that it can write it to the punch machine file. The intermediate file is opened and phase two begins. Two variables are used, Xo and Xn, for the previous X-Y coordinate and the current X-Y coordinate just read from the intermediate file.

```
start2 :-
        see(gerberint),
        Xloc is 0,
        Yloc is 0,
        Xo = x(Xloc,Yloc),
        trans_gerint(Xo).
```

The X-Y coordinates are read from the intermediate file and passed to a series of rules along with the previous coordinate.

```
trans_gerint(Xo) :-
        read(Xn),
        write_coord(Xn,Xo).
```

The following Prolog rule **write_coord/2** signifies "end-of-file". The translation program closes all files and halts.

```
write_coord(m(02),_Xo) :-
        told,
        halt.
```

The following rules of **write_coord/2** extract the X-Y coordinates from the intermediate form. There are several variations of gerber words in the intermediate form. When an X or Y coordinate is not repeated in the intermediate file, the previous coordinate is used from the Xo coordinate. The new coordinates are divided by 10,000. This is to format the coordinates in inches and in floating point format as required by the CNC Punch Machine.

```
write_coord(x(Xloc,Yloc,03),_Xo) :-
        Xloc1 is Xloc/10000,
        write_xy_loc(Xloc1),
        write(' '),
        Yloc1 is Yloc/10000,
        write_xy_loc(Yloc1),
        nl,
        trans_gerint(x(Xloc1,Yloc1)),
        !.

write_coord(y(Yloc,03),x(Xloco,_Yloco)) :-
```

```
        write_xy_loc(Xloco),
        write(' '),
        Yloc1 is Yloc/10000,
        write_xy_loc(Yloc1),
        nl,
        trans_gerint(x(Xloco,Yloc1)),
        !.

write_coord(x(Xloc,03),x(_Xloco,Yloco)) :-
        Xloc1 is Xloc/10000,
        write_xy_loc(Xloc1),
        write(' '),
        write_xy_loc(Yloco),
        nl,
        trans_gerint(x(Xloc1,Yloco)),
        !.

write_coord(d(03),x(Xloco,Yloco)) :-
        write_xy_loc(Xloco),
        write(' '),
        write_xy_loc(Yloco),
        nl,
        trans_gerint(x(Xloco,Yloco)),
        !.
```

A gerber file is used for much more than punching or drilling holes, it is a complete photoplotting format. For hole punching, only the "D03" code is used or needed. The following rules allow the translator to continue on when other "D" codes exist in the gerber file. The translator keeps track of the coordinates where another "D" code is used and continues on. This enables the translator to be more versatile for hole punching. Any gerber file can be read for translation and only the "D03" codes will be translated to the CNC Punch files.

```
write_coord(x(Xloc,Yloc,_Tool),_Xo) :-
        Xloc1 is Xloc/10000,
        Yloc1 is Yloc/10000,
        trans_gerint(x(Xloc1,Yloc1)),
        !.

write_coord(y(Yloc,_Tool),x(Xloco,_Yloco)) :-
        Yloc1 is Yloc/10000,
        trans_gerint(x(Xloco,Yloc1)),
        !.

write_coord(x(Xloc,_Tool),x(_Xloco,Yloco)) :-
        Xloc1 is Xloc/10000,
        trans_gerint(x(Xloc1,Yloco)),
```

```
        !.

write_coord(d(_Tool),Xo) :-
        trans_gerint(Xo),
        !.

write_coord(x(Xloc,Yloc),_Xo) :-
        Xloc1 is Xloc/10000,
        Yloc1 is Yloc/10000,
        trans_gerint(x(Xloc1,Yloc1)),
        !.

write_coord(x(Xloc),x(_Xloco,Yloco)) :-
        Xloc1 is Xloc/10000,
        trans_gerint(x(Xloc1,Yloco)),
        !.

write_coord(y(Yloc),x(Xloco,_Yloco)) :-
        Yloc1 is Yloc/10000,
        trans_gerint(x(Xloco,Yloc1)),
        !.
```

A select tool command G54 is detected here. The translator closes the previous file of X-Y coordinates and opens a new file for the next block of coordinates. The user is prompted for file names and how the filename should be entered. Single quotes around the file name are necessary because of the significance of the period to Prolog. Prolog uses the period to end a rule or series of rules. A filename with an ".ext" violates Prolog's use of the period. Putting single quotes around the filename turns the filename into an atom which Prolog can use.

```
write_coord(g(54,Tool),Xo) :-
        told, tell(user),
        write('[ Enter the file name for the tool d'),
        write(Tool),
        write(', ''file.ext''.  ]'),
        nl,
        see(user),
        read(Filename),
        tell(Filename),
        see(gerberint),
        trans_gerint(Xo),
        !.
```

When a "G" code other than G54 is encountered, the translator skips over the "G" code in the intermediate file and continues on. A G54 code is the only "G" code that has significance to the CNC Punch Machine.

```
write_coord(g(_),Xo) :-
        trans_gerint(Xo),
        !.
```

The X-Y locations are written in the correct format for the CNC Punch Machine. If the coordinate is greater than zero, a "+" sign is written to the file followed by the coordinate. The coordinate must be written with four numbers after the decimal point, regardless of the numbers being zero or not. If the number is less than zero, then the coordinate is just written in the correct format. Since the number is already negative there is no need to insert a sign.

```
write_xy_loc(X_Yloc) :-
        X_Yloc >= 0,
        write('+'),
        format('~4f', X_Yloc),
        !.

write_xy_loc(X_Yloc) :-
        format('~4f', X_Yloc).
```

5.0 Example of an MCM Design

Presented here is an example of the Gerber Format to CNC Punch Translator being used to aid in the making of an MCM. Shown in Figure 1A is an MCM design with four vias, spaced 0.30 inch apart from each other in a square fashion. The origin of this particular MCM is in the lower left corner, with the X-Y coordinate (0,0). Shown in Figure 1B is the gerber file which was generated, by Intergraph, to photoplot the MCM artwork. Typically here at EPSD, the Intergraph CAD system is used as the first step in making an MCM. The signal traces and vias are drawn in Intergraph and a gerber file is generated for photoplotting the artwork.

5.1 Translating to the Intermediate Form

Translation from Gerber Format to a CNC Punch Machine file is begun by executing the translator with the command "gerber_trans". Upon execution, the user is prompted for the name of the gerber file and as to how the filename should be entered. Upon entering the filename, the translator begins mapping the gerber file into a Prolog intermediate form. Each gerber word in the gerber file is represented in the intermediate form as a Prolog structure which is used by phase two of the translator. Shown in Figure 1C is the intermediate file for the MCM design.

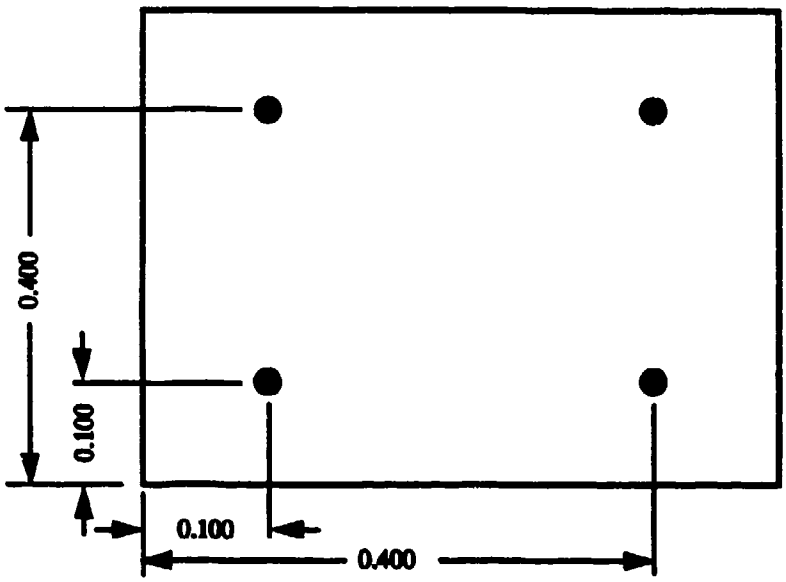

Figure 1A. MCM Drawing
(dimensions are in inches)

```
G54D19*
X1000Y1000D03*
Y4000D03:
X4000Y1000D03*
Y4000D03*
M02*
```

Figure 1B. Gerber File

```
g(54,19).
x(1000,1000,03).
y(4000,03).
x(4000,1000,03).
y(4000,03).
m(02).
```

Figure 1C. Intermediate File

Figure 1. Example of an MCM Design

5.2 Creating the CNC Punch File

When the gerber file is completely read, the user is prompted for an output filename for the X-Y coordinates. The translator then begins reading the intermediate file and writes an X-Y coordinate file in CNC Punch Machine format. An X and Y coordinate must exist for each hole to be punched. Notice in the above intermediate file, an X coordinate does not exist on line two. Only a Y coordinate exists because the X coordinate did not change for the next hole. Therefore the translator must keep track of the previous X-Y coordinate on line one so that an X coordinate can be written to the CNC Punch Machine file for the hole on line two. The same is repeated for line four where again there is no X coordinate in the intermediate file. Finally, the coordinate is divided by 10,000 and written to the X-Y coordinate CNC Punch file. As the coordinate is written, it is formatted to a signed floating point number with four places after the decimal. Shown in Figure 2 is the X-Y file for the CNC Punch Machine after translation.

```
+0.1000 +0.1000
+0.1000 +0.4000
+0.4000 +0.1000
+0.4000 +0.4000
```

Figure 2. X-Y Coordinate File for CNC Punch Machine

6.0 Conclusion

In designing the Gerber to CNC Punch Translator, the process of programming the CNC Punch Machine is now fully automated. Previously, a user had to program the X and Y coordinates for each hole to be punched by hand. Now a gerber file, which is a standard format for photoplotting and used by most CAD systems, is easily used to program the CNC Punch Machine. Given that the translator is designed to read any gerber file, formatted in absolute mode and in inches, it is not necessary to strip out the gerber control words, from the gerber file, which do not pertain to hole punching. The Gerber Format to CNC Punch Translator simply extracts the via coordinates from the gerber file and generates an X-Y coordinate file. Thus, a gerber file which was created to photoplot an MCM design, including signal traces as well as via locations, is readily usable to program the CNC Punch Machine.

7.0 References

Quintus Computer Systems, Incorporated, *Quintus Prolog Reference Manual*, Mountain View, California, 1988.

Clocksin, W.F. and Mellish, C.S. *Programming in Prolog*. New York: Springer-Verlag, 1987.

Gerber Systems Corporation, *Gerber Format, Plot Data Format Reference Book*, South Windsor, Connecticut, 1993.

ARMY RESEARCH LABORATORY
ELECTRONICS AND POWER SOURCES DIRECTORATE
CONTRACT OR IN-HOUSE TECHNICAL REPORT
MANDATORY DISTRIBUTION LIST

June 1994
Page 1 of 2

Defense Technical Information Center*
ATTN: DTIC-OCC
Cameron Station (Bldg 5)
Alexandria, VA 22304-6145
(*Note: Two copies will be sent from STINFO office, Fort Monmouth, NJ)

Director
US Army Material Systems Analysis Actv
ATTN: DRXSY-MP
(1) Aberdeen Proving Ground, MD 21005

Commander, AMC
ATTN: AMCDE-SC
5001 Eisenhower Ave.
(1) Alexandria, VA 22333-0001

Director
Army Research Laboratory
ATTN: AMSRL-D (John W. Lyons)
2800 Powder Mill Road
(1) Adelphi, MD 20783-1145

Director
Army Research Laboratory
ATTN: AMSRL-DD (COL William J. Miller)
2800 Powder Mill Road
(1) Adelphi, MD 20783-1145

Director
Army Research Laboratory
2800 Powder Mill Road
Adelphi, MD 20783-1145
(1) AMSRL-OP-CI-AD (Tech Pubs)
(1) AMSRL-OP-CI-AD (Records Mgt)
(1) AMSRL-OP-CI-AD (Tech Library)

Directorate Executive
Army Research Laboratory
Electronics and Power Sources Directorate
Fort Monmouth, NJ 07703-5601
(1) AMSRL-EP
(1) AMSRL-EP-T (M. Howard)
(1) AMSRL-OP-RM-FH
(22) Originating Office

Advisory Group on Electron Devices
ATTN: Documents
2011 Crystal Drive, Suite 307
(2) Arlington, VA 22202

Commander, CECOM
R&D Technical Library
Fort Monmouth, NJ 07703-5703
(1) AMSEL-IM-BM-I-L-R (Tech Library)
(3) AMSEL-IM-BM-I-L-R (STINFO ofc)

Deputy for Science & Technology
Office, Asst Sec Army (R&D)
(1) Washington, DC 20310

HQDA (DAMA-ARZ-D/
Dr. F.D. Verderame)
(1) Washington, DC 20310

Director
Naval Research Laboratory
ATTN: Code 2627
(1) Washington, DC 20375-5000

Cdr, PM JTFUSION
ATTN: JTF
1500 Planning Research Drive
(1) McLean, VA 22102

Rome Air Development Center
ATTN: Documents Library (TILD)
(1) Griffiss AFB, NY 13441

Dir, ARL Battlefield
Environment Directorate
ATTN: AMSRL-BE
White Sands Missile Range
(1) NM 88002-3501

Dir, ARL Sensors, Signatures,
Signal & Information Processing
Directorate (S3I)
ATTN: AMSRL-SS
2800 Powder Mill Road
(1) Adelphi, MD 20783-1145

Dir, CECOM Night Vision/
Electronic Sensors Directorate
ATTN: AMSEL-RD-NV-D
(1) Fort Belvoir, VA 22060-5677

Dir, CECOM Intelligence and
Electronic Warfare Directorate
ATTN: AMSEL-RD-IEW-D
Vint Hill Farms Station
(1) Warrenton, VA 22186-5100

Cdr, Marine Corps Liaison Office
ATTN: AMSEL-LN-MC
(1) Fort Monmouth, NJ 07703-5033